**《软件逆向工程》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | | **2022级信息安全** | | | **姓名** |  |
| **实验题目** | PE文件结构分析 | | | | | |
| **实验时间** | **2024/10/18** | | **实验地点** | **DS3401** | | |
| **实验成绩** |  | | **实验性质** | **□验证性 □设计性 ☑综合性** | | |
| 教师评价：  ☑算法/实验过程正确； ☑源程序/实验内容提交； ☑程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； ☑报告规范；  其他：  评价教师签名： 杨吉云 | | | | | | |
| 1. 实验目的 2. 验证局部变量的机制 3. 验证PE文件头结构 4. 验证PE文件节表结构 | | | | | | |
| 1. 实验项目内容 2. 编写程序，要求：1）在数据区定义10个单元的word数组，初始化为确定的数值；2）含1个子程序，在子程序中定义10个单元的word数组，将数据区定义的数组赋给子程序定义的数组；在主程序中调用子程序，记录堆栈变化情况。 3. 在1编写程序的基础上，加入调用MessageBox函数生成exe文件，利用Winhex、OD分析该exe文件的结构； | | | | | | |
| **三、实验过程或算法（源程序）**  .386  .model flat,stdcall  option casemap:none  .stack 4096  include windows.inc  include user32.inc  includelib user32.lib  includelib kernel32.lib  includelib masm32.lib  ExitProcess PROTO, dwExitCode:DWORD  .data  arr word 1h, 2h, 3h, 4h, 5h, 6h, 7h, 8h, 9h, 0Ah  msg db 'Hello! I am Yao Fan.', 0  .code  COPY proc  push esi  push ecx  push ebp  mov ebp,esp  sub edi,20  mov edi,ebp    L1:  mov ax, [esi]  mov WORD PTR [edi], ax  add esi, TYPE WORD  add edi, TYPE WORD  loop L1    pop ebp  pop ecx  pop esi  ret  COPY endp  main proc  invoke MessageBox,NULL,offset msg, NULL, MB\_OK  mov esi, offset arr  mov ecx, lengthof arr  call COPY  invoke ExitProcess,NULL  ret  main endp  end main | | | | | | |
| 四、实验结果及分析和（或）源程序调试过程  **1.messageBox输出**  运行后messageBOX弹窗如图：    **2.堆栈状态分析**  初始堆栈状态    调用子过程    压入esi、ecx、ebp    数组初始数值压入栈中    **3.PE文件分析**  首先使用LordPE打开lab3.exe文件，可以观察到这个PE文件的参数如下图：  C:\RadASM\Masm\Projects\lab3\lab3.exe    后续分析PE文件结构的时候可以与LordPE中分析出来的数据进行对比验证。    然后使用WinHex打开lab3.exe文件进行分析，此时PE文件还没有被运行到进程里，是在文件状态下，可以看到此时WinHex左侧的offset是从00000000开始计算的，表示数据距离文件头的偏移量RVA。  首先找到exe标志字“MZ”即 5A4D 处，在MZ header之后是 DOS stub，这一部分是DOS文件结构的保留内容不做分析  接下来找到PE文件头的RVA，在DOS\_HEADER结构中对应的是e\_lfanew  的值，e\_lfanew位于文件头偏移3C字节处，如下图，可以得到标准PE头是从00B0处开始的。    定位到左侧offset值000000B0处，从这里开始首先是PE头标识  signature，它的值是固定的00004550。从这里开始往后20个字节是  FILE\_HEADER，根据下图可以得到：    PE中节的数量： 3h  PE文件创建的日期和时间：67124D9C    SizeOfOptionalHeader：00E0    FILE\_HEADER后接着的是OPTIONAL\_HEADER（下图紫色标深部分），从标准PE头偏移量0018h处开始（也就是距文件头偏移00C8h处）开始。    这里可以分析出：  SizeOfCode: 200h  SizeOfInitializedData: 400h  AddressOfEntryPoint: 101Ch    BaseOfCode：0001000  BaseOfData：00002000    ImageBase：00400000    内存对齐粒度： 1000h （符合32位系统默认值）  文件对齐力度： 200h    SizeOfImage:00003029    SizeOfHeaders:00000400    校验和：000BBA4    OPTIONAL\_HEADER最后一个字段是数据目录项DataDirectory,从标准PE头偏移量0078h处开始    由于代码中只调用了一个messagebox函数，所以可以看到数据目录中只有导入表的数据  导入表RVA： 2010h  导入表大小： 3Ch  OPTIONAL\_HEADER后面接着的是节表，从lordPE中可以看到我们这个文件中有3个节表项，分别是text、rdata、data，如下图所示：    接下来在WinHex中进行分析，节表项是跟在optionalheader后边  的  .text节表：    根据节表项的数据结构可以得到text节表的：  VirtualSize：50h  RVA：1000h  --这里查看ollyICE可以看到我们的命令确实是从00401000处开始存放的    在文件中的偏移：00000400  在文件中对齐后的尺寸： 200h  在text节后边的是rdata节表：    根据节表项的数据结构可以得到rdata节表的：  VirtualSize:92h  RVA：2000h  在文件中的偏移： 600h  在文件中对齐后的尺寸：200h  在rdata节后边的是data节表：    根据节表项的数据结构可以得到data节表的：  VirtualSize:22h  RVA：00003000  --这里查看ollyICE可以看到我们初始定义的数据确实是从00403000处开始存放的    在文件中的偏移： 800h  --在800h的地方可以看到输出的数据    在文件中对齐后的尺寸：200h  由于导入表的RVA是2010h，text节最多到1050h，rdata节最多到2092h，所以导入表在rdata节中，可以计算导入表的文件虚拟地址  **2010h-2000h+600h=610h**  根据导入表在文件中的偏移地址可以找到它的位置，如下图所示：    OriginalFirstThunk：2054h  --对应的文件中的偏移地址是654h，指向的数据位205Ch  --205C指向的位置（在文件中就是65Ch）就是被调用的函数或者动态链接库的数据，如上图双框区域标注所示。  FirstThunk：2008h  --2008h在文件中指向的就是608h的位置的值，即205Ch。这里是因  为PE文件还没有进入进程状态，被调用函数的虚拟地址还没有被写入FirstThunk中，所以根据默认的方式，会将OriginalFirstThunk对应的值复制到FirstThunk中。  最后对PE文件进行运行：    运行过后发现FirstThunk的值变为75D3B730h，说明被导入函数的虚拟地址已被导入到导入表中 | | | | | | |